Package ‘cNORM’

July 19, 2024

Type Package

Title Continuous Norming

Version 3.1.0

Maintainer Wolfgang Lenhard <wolfgang.lenhard@uni-wuerzburg.de>

Date 2024-07-19

Description Conventional methods for producing standard scores or percentiles in psychometrics or biometrics are often plagued with 'jumps' or 'gaps' (i.e., discontinuities) in norm tables and low confidence for assessing extreme scores. The continuous norming method introduced by A. Lenhard et al. (2016, <doi:10.1177/1073191116656437>; 2019, <doi:10.1371/journal.pone.0222279>; 2021 <doi:10.1177/0013164420928457>) estimates percentile development (e. g. over age) and generates continuous test norm scores on the basis of the raw data from standardization samples, without requiring assumptions about the distribution of the raw data: Norm scores are directly established from raw data by modeling the latter ones as a function of both percentile scores and an explanatory variable (e.g., age). The method minimizes bias arising from sampling and measurement error, while handling marked deviations from normality, addressing bottom or ceiling effects and capturing almost all of the variance in the original norm data sample. It includes procedures for post stratification of norm samples to overcome bias in data collection and to mitigate violations of representativeness. An online demonstration is available via <https://cnorm.shinyapps.io/cNORM/>.

Depends R (>= 4.0.0)

Imports lattice (>= 0.21), leaps (>= 3.1), latticeExtra (>= 0.6)

Suggests knitr, shiny, shinycssloaders, foreign, readxl, rmarkdown, testthat

License AGPL-3

VignetteBuilder knitr

RoxygenNote 7.3.0

NeedsCompilation no

Repository CRAN

Encoding UTF-8

LazyData true

URL https://www.psychometrica.de/cNorm_en.html,
https://github.com/WLenhard/cNORM

BugReports https://github.com/WLenhard/cNORM/issues

Author Alexandra Lenhard [aut] (<https://orcid.org/0000-0001-8680-4381>),
Wolfgang Lenhard [cre, aut] (<https://orcid.org/0000-0002-8184-6889>),
Sebastian Gary [aut],
WPS publisher [fnd] (<https://www.wpspublish.com/>)

Date/Publication 2024-07-19 08:00:01 UTC

Contents

bestModel . . . 3
betaByGroup . . . 5
betaCoefficients . . . 6
betaContinuous . . . 7
betaNormTable . . . 8
betaTable . . . 9
buildFunction . . . 10
calcPolyInL . . . 10
calcPolyInLBase . . . 11
calcPolyInLBase2 . . . 11
CDC . . . 12
checkConsistency . . . 13
checkWeights . . . 14
cnorm . . . 15
cnorm.cv . . . 17
cNORM.GUI . . . 20
computePowers . . . 20
computeWeights . . . 22
derivationTable . . . 23
derive . . . 24
elfe . . . 25
epm . . . 26
getGroups . . . 27
getNormCurve . . . 28
getNormScoreSE . . . 29
life . . . 30
modelSummary . . . 31
mortality . . . 31
normTable . . . 32
plot.cnorm . . . 34
plotCnorm . . . 35
plotDensity . . . 35
plotDerivative . . . 36

plotNorm . . . 37
plotNormCurves . . . 38
plotPercentiles . . . 40
plotPercentileSeries . . . 41
plotRaw . . . 42
plotSubset . . . 43
ppvt . . . 44
predictBeta . . . 45
predictNorm . . . 46
predictRaw . . . 47
prepareData . . . 48
prettyPrint . . . 50
print.cnorm . . . 51
printSubset . . . 51
rangeCheck . . . 52
rankByGroup . . . 53
rankBySlidingWindow . . . 55
rawTable . . . 57
regressionFunction . . . 59
simMean . . . 60
simSD . . . 60
simulateRasch . . . 61
standardizeRakingWeights . . . 62
summary.cnorm . . . 63
weighted.quantile . . . 64
weighted.quantile.harrell.davis . . . 65
weighted.quantile.inflation . . . 65
weighted.quantile.type7 . . . 66
weighted.rank . . . 67

Index **68**

`bestModel` *Best-fitting Regression Model Based on Powers and Interactions*

Description

Computes and selects the best-fitting regression model by evaluating a series of models with increasing predictors. It aims to find a parsimonious model that effectively captures the variance in the data. This can be useful in psychometric test construction to smooth out data and reduce noise while retaining key diagnostic information. Model selection can be based on the number of terms or the explained variance (R^2). Setting high values for the number of terms, R^2 cutoff, or ‘k‘ may lead to overfitting. Typical recommended starting points are ‘terms = 5‘, ‘R^2 = .99‘, and ‘k = 4‘.

Usage

```
bestModel(
  data,
  raw = NULL,
  R2 = NULL,
  k = NULL,
  t = NULL,
  predictors = NULL,
  terms = 0,
  weights = NULL,
  force.in = NULL,
  plot = TRUE
)
```

Arguments

`data`	Preprocessed dataset with 'raw' scores, powers, interactions, and usually an explanatory variable (like age).
`raw`	Name of the raw score variable (default: 'raw').
`R2`	Adjusted R^2 stopping criterion for model building (default: 0.99).
`k`	Power constant influencing model complexity (default: 4, max: 6).
`t`	Age power parameter. If unset, defaults to 'k'.
`predictors`	List of predictors or regression formula for model selection. Overrides 'k' and can include additional variables.
`terms`	Desired number of terms in the model.
`weights`	Optional case weights. If set to FALSE, default weights (if any) are ignored.
`force.in`	Variables forcibly included in the regression.
`plot`	If TRUE (default), displays a percentile plot of the model.

Details

Additional functions like `plotSubset(model)` and `cnorm.cv` can aid in model evaluation.

Value

The model meeting the R^2 criteria. Further exploration can be done using `plotSubset(model)` and `plotPercentiles(data, model)`.

See Also

plotSubset, plotPercentiles, plotPercentileSeries, checkConsistency

Other model: `checkConsistency()`, `cnorm.cv()`, `derive()`, `modelSummary()`, `print.cnorm()`, `printSubset()`, `rangeCheck()`, `regressionFunction()`, `summary.cnorm()`

Examples

```
## Not run:
# Example with sample data
normData <- prepareData(elfe)
model <- bestModel(normData)
plotSubset(model)
plotPercentiles(normData, model)

# Specifying variables explicitly
preselectedModel <- bestModel(normData, predictors = c("L1", "L3", "L1A3", "A2", "A3"))
print(regressionFunction(preselectedModel))

# Modeling based on the CDC data
bmi.data <- prepareData(CDC, raw = "bmi", group = "group", age = "age")
bmi.model <- bestModel(bmi.data, raw = "bmi")
printSubset(bmi.model)

# Using a precomputed model formula for gender-specific models
bmi.model.boys <- bestModel(bmi.data[bmi.data$sex == 1, ], predictors = bmi.model$terms)
bmi.model.girls <- bestModel(bmi.data[bmi.data$sex == 2, ], predictors = bmi.model$terms)

# Using a custom list of predictors and incorporating the 'sex' variable
bmi.sex <- bestModel(bmi.data, raw = "bmi", predictors = c(
  "L1", "L3", "A3", "L1A1", "L1A2", "L1A3", "L2A1", "L2A2",
  "L2A3", "L3A1", "L3A2", "L3A3", "sex", force.in = c("sex"))

## End(Not run)
```

---

betaByGroup — *Estimate Beta-Binomial Parameters by Group*

Description

This function calculates the beta-binomial distribution parameters (alpha, beta, mean, variance) for subsets of data grouped by a specified factor. It applies the ‘betaCoefficients‘ function to each group separately, aggregating the results into a single data frame. This is particularly useful for analyzing heterogeneity in success probabilities across different groups within a dataset.

Usage

```
betaByGroup(x, group, n)
```

Arguments

x	A vector of non-negative integers representing the number of successes in trials for the entire dataset.
group	A factor or similar object that divides ‘x‘ into groups. Each element of ‘x‘ is associated with a group indicated by the corresponding element in ‘group‘.
n	The maximum number of trials, assumed to be the same for all groups.

Details

The function first identifies unique groups in the ‘group‘ argument and then iterates over these groups. For each group, it extracts the subset of ‘x‘ corresponding to that group and computes the beta-binomial distribution parameters using the ‘betaCoefficients‘ function. The results are compiled into a matrix that is then converted into a data frame for easier manipulation and interpretation.

Value

A data frame where each row contains the beta-binomial distribution parameters (alpha ‘a‘, beta ‘b‘, mean ‘m‘, variance ‘var‘) for a group, along with the group identifier. The columns are named ‘a‘, ‘b‘, ‘m‘, ‘var‘, ‘n‘, and ‘group‘, with each row corresponding to a distinct group in the input.

Examples

```
x <- elfe$raw
group <- elfe$group
n <- 26
betaByGroup(x, group, n)
```

`betaCoefficients` *Compute Parameters of a Beta Binomial Distribution*

Description

This function calculates the α (a) and β (b) parameters of a beta binomial distribution, along with the mean (m), variance (var) based on the input vector ‘x‘ and the maximum number ‘n‘.

Usage

```
betaCoefficients(x, n)
```

Arguments

`x`	A numeric vector of non-negative integers representing observed counts.
`n`	The maximum number or the maximum possible value of ‘x‘.

Details

The beta-binomial distribution is a discrete probability distribution that models the number of successes in a fixed number of trials, where the probability of success varies from trial to trial. This variability in success probability is modeled by a beta distribution. Such a calculation is particularly relevant in scenarios where there is heterogeneity in success probabilities across trials, which is common in real-world situations, as for example the number of correct solutions in a psychometric test, where the test has a fixed number of items.

Value

A numeric vector containing the calculated parameters in the following order: alpha (a), beta (b), mean (m), variance (var), and the maximum number (n).

Examples

```
x <- c(1, 2, 3, 4, 5)
n <- 5
betaCoefficients(x, n)
```

`betaContinuous` *Continuous Norming with Beta-Binomial Distribution (experimental)*

Description

This function models the alpha (‘a‘) and beta (‘b‘) parameters of the beta-binomial distribution across groups using polynomial regression. It then calculates the distribution’s properties (cumulative probabilities, density, percentiles, and z-scores) for these modeled parameters. The modeling of ‘a‘ and ‘b‘ allows for the investigation of how these parameters vary with a continuous group variable, allowing for continuous norming.

Usage

```
betaContinuous(param, powerA = Inf, powerB = Inf)
```

Arguments

`param`	A data frame containing the columns ‘a‘, ‘b‘, ‘group‘, and ‘n‘. Each row should represent a distinct group with its corresponding beta-binomial parameters and the group identifier. These parameters can be obtained with the ’betaByGroup’ function.
`powerA`	The degree of the polynomial used to model the ‘a‘ parameter across groups. Please choose $powerA \leq k$ with k being the number of groups.
`powerB`	The degree of the polynomial used to model the ‘b‘ parameter across groups. Please choose $powerB \leq k$ with k being the number of groups.

Details

The function first fits polynomial regression models for ‘a‘ and ‘b‘ against a continuous group variable, allowing for non-linear trends in how the shape parameters of the beta-binomial distribution change with the group. It then predicts ‘a‘ and ‘b‘ for each group, using these predicted values to calculate the beta-binomial distribution’s properties for each group. This approach facilitates understanding the variability and dynamics of the distribution across different conditions or groups.

Value

A list containing several components: ‘manifestParameters‘ with the input parameters, ‘powerA‘ and ‘powerB‘ showing the polynomial degrees used, ‘modA‘ and ‘modB‘ with the polynomial regression models for ‘a‘ and ‘b‘ parameters.

Examples

```
param <- data.frame(a = c(1,2,3), b = c(2,3,4), group = c(1,2,3), n = c(30,30,30))
powerA <- 2
powerB <- 2
betaContinuous(param, powerA, powerB)
```

---

betaNormTable — *Generate norm table from parametric continuous norming with Beta-Binomial Parameters*

Description

This function generates a table of beta-binomial distribution properties (cumulative probabilities, density, percentiles, and z-scores) for a specified group, using alpha (‘a‘) and beta (‘b‘) parameters predicted by a model created with the ‘betaContinuous‘ function.

Usage

```
betaNormTable(model, group, m = NULL)
```

Arguments

model	A list containing the components from a ‘betaContinuous‘ model output.
group	A number specifying the group variable for which predictions and subsequent beta-binomial distribution calculations are desired.
m	An optional stop criterion in table generation. Positive integer lower than n

Value

A data frame with columns representing the number of successes (‘x‘), the probability mass function values (‘Px‘), cumulative probabilities (‘Pcum‘), percentiles (‘Percentile‘), and z-scores (‘z‘) for the specified group based on the predicted ‘a‘ and ‘b‘ parameters.

Examples

```
# Determies beta parameters and models these continuously
param <- betaByGroup(elfe$raw, elfe$group, 26)
beta.model <- betaContinuous(param, 4, 4)

# Calculates table for new group
newGroup <- 3.9
betaNormTable(beta.model, newGroup)
```

betaTable — *Calculate Cumulative Probabilities, Density, Percentiles, and Z-Scores for Beta-Binomial Distribution*

Description

This function computes the cumulative probabilities, the density (probability mass function values), the percentiles, and the corresponding z-scores based on the specified parameters of a beta-binomial distribution. The beta-binomial distribution is used to model the number of successes in a fixed number of trials with success probability varying from trial to trial, described by beta distribution parameters α (alpha) and β (beta).

Usage

```
betaTable(a, b, n, m = NULL)
```

Arguments

a	The α parameter of the beta distribution, indicating the shape parameter associated with successes.
b	The β parameter of the beta distribution, indicating the shape parameter associated with failures.
n	The number of trials in the beta-binomial distribution.
m	An optional stop criterion in table generation. Positive integer lower than n.

Details

The function utilizes the gamma function to calculate factorial terms needed for the probability mass function (PMF) and cumulative distribution function (CDF) calculations of the beta-binomial distribution. It iterates over the range of possible successes (0 to n) to compute the PMF values (Px), cumulative probabilities ($Pcum$), and mid-percentiles. These percentiles are then used to calculate the corresponding z-scores, which indicate how many standard deviations an element is from the mean.

Value

A data frame with columns:

x The number of successes (0 to n).

Px The density (probability mass function value) for each number of successes.

Pcum The cumulative probability up to each number of successes.

Percentile The percentile corresponding to each number of successes.

z The z-score corresponding to each percentile.

Examples

```
betaTable(2, 2, 45, 20)
```

`buildFunction` *Build regression function for bestModel*

Description

Build regression function for bestModel

Usage

```
buildFunction(raw, k, t, age, covariates)
```

Arguments

`raw`	name of the raw score variable
`k`	the power degree for location
`t`	the power degree for age
`age`	use age
`covariates`	use covariates

Value

reression function

`calcPolyInL` *Internal function for retrieving regression function coefficients at specific age*

Description

The function is an inline for searching zeros in the inverse regression function. It collapses the regression function at a specific age and simplifies the coefficients.

Usage

```
calcPolyInL(raw, age, model)
```

Arguments

`raw`	The raw value (subtracted from the intercept)
`age`	The age
`model`	The cNORM regression model

Value

The coefficients

`calcPolyInLBase` *Internal function for retrieving regression function coefficients at specific age*

Description

The function is an inline for searching zeros in the inverse regression function. It collapses the regression function at a specific age and simplifies the coefficients.

Usage

```
calcPolyInLBase(raw, age, coeff, k)
```

Arguments

`raw`	The raw value (subtracted from the intercept)
`age`	The age
`coeff`	The cNORM regression model coefficients
`k`	The cNORM regression model power parameter

Value

The coefficients

`calcPolyInLBase2` *Internal function for retrieving regression function coefficients at specific age (optimized)*

Description

The function is an inline for searching zeros in the inverse regression function. It collapses the regression function at a specific age and simplifies the coefficients. Optimized version of the prior 'calcPolyInLBase'

Usage

```
calcPolyInLBase2(raw, age, coeff, k)
```

Arguments

`raw`	The raw value (subtracted from the intercept)
`age`	The age
`coeff`	The cNORM regression model coefficients
`k`	The cNORM regression model power parameter

Value

The coefficients

`CDC` *BMI growth curves from age 2 to 25*

Description

By the courtesy of the Center of Disease Control (CDC), cNORM includes human growth data for children and adolescents age 2 to 25 that can be used to model trajectories of the body mass index and to estimate percentiles for clinical definitions of under- and overweight. The data stems from the NHANES surveys in the US and was published in 2012 as public domain. The data was cleaned by removing missing values and it includes the following variables from or based on the original dataset.

Usage

```
CDC
```

Format

A data frame with 45053 rows and 7 variables:

age continuous age in years, based on the month variable

group age group; chronological age in years at the time of examination

month chronological age in month at the time of examination

sex sex of the participant, 1 = male, 2 = female

height height of the participants in cm

weight weight of the participants in kg

bmi the body mass index, computed by (weight in kg)/(height in m)^2

A data frame with 45035 rows and 7 columns

Source

https://www.cdc.gov/nchs/nhanes/index.htm

References

CDC (2012). National Health and Nutrition Examination Survey: Questionnaires, Datasets and Related Documentation. available https://www.cdc.gov/nchs/nhanes/index.htm (date of retrieval: 25/08/2018)

`checkConsistency` *Check the consistency of the norm data model*

Description

While abilities increase and decline over age, within one age group, the norm scores always have to show a linear increase or decrease with increasing raw scores. Violations of this assumption are a strong indication for problems in modeling the relationship between raw and norm scores. There are several reasons, why this might occur:

1. Vertical extrapolation: Choosing extreme norm scores, e. g. values -3 <= x and x >= 3 In order to model these extreme values, a large sample dataset is necessary.
2. Horizontal extrapolation: Taylor polynomials converge in a certain radius. Using the model values outside the original dataset may lead to inconsistent results.
3. The data cannot be modeled with Taylor polynomials, or you need another power parameter (k) or R2 for the model.

In general, extrapolation (point 1 and 2) can carefully be done to a certain degree outside the original sample, but it should in general be handled with caution.

Usage

```
checkConsistency(
  model,
  minAge = NULL,
  maxAge = NULL,
  minNorm = NULL,
  maxNorm = NULL,
  minRaw = NULL,
  maxRaw = NULL,
  stepAge = 1,
  stepNorm = 1,
  warn = FALSE,
  silent = FALSE,
  covariate = NULL
)
```

Arguments

`model`	The model from the bestModel function or a cnorm object
`minAge`	Age to start with checking
`maxAge`	Upper end of the age check
`minNorm`	Lower end of the norm value range
`maxNorm`	Upper end of the norm value range
`minRaw`	clipping parameter for the lower bound of raw scores

maxRaw	clipping parameter for the upper bound of raw scores
stepAge	Stepping parameter for the age check, usually 1 or 0.1; lower values indicate higher precision / closer checks
stepNorm	Stepping parameter for the norm table check within age with lower scores indicating a higher precision. The choice depends of the norm scale used. With T scores a stepping parameter of 1 is suitable
warn	If set to TRUE, already minor violations of the model assumptions are displayed (default = FALSE)
silent	turn off messages
covariate	In case, a covariate has been used, please specify the degree of the covariate / the specific value here.

Value

Boolean, indicating model violations (TRUE) or no problems (FALSE)

See Also

Other model: bestModel(), cnorm.cv(), derive(), modelSummary(), print.cnorm(), printSubset(), rangeCheck(), regressionFunction(), summary.cnorm()

Examples

```
result <- cnorm(raw = elfe$raw, group = elfe$group)
modelViolations <- checkConsistency(result,
  minAge = 2, maxAge = 5, stepAge = 0.1,
  minNorm = 25, maxNorm = 75, minRaw = 0, maxRaw = 28, stepNorm = 1
)
plotDerivative(result, minAge = 2, maxAge = 5, minNorm = 25, maxNorm = 75)
```

---

checkWeights *Check, if NA or values <= 0 occur and issue warning*

Description

Check, if NA or values <= 0 occur and issue warning

Usage

```
checkWeights(weights)
```

Arguments

weights	Raking weights

cnorm *Continuous Norming*

Description

Conducts continuous norming in one step and returns an object including ranked raw data and the continuous norming model. Please consult the function description ' of 'rankByGroup', 'rankBySlidingWindow' and 'bestModel' for specifics of the steps in the data preparation and modeling process. In addition to the raw scores, either provide

- a numeric vector for the grouping information (group)
- a numeric age vector and the width of the sliding window (age, width)

for the ranking of the raw scores. You can adjust the grade of smoothing of the regression model by setting the k and terms parameter. In general, increasing k to more than 4 and the number of terms lead to a higher fit, while lower values lead to more smoothing. The power parameter for the age trajectory can be specified independently by 't'. If both parameters are missing, cnorm uses k = 5 and t = 3 by default.

Usage

```
cnorm(
  raw = NULL,
  group = NULL,
  age = NULL,
  width = NA,
  weights = NULL,
  scale = "T",
  method = 4,
  descend = FALSE,
  k = NULL,
  t = NULL,
  terms = 0,
  R2 = NULL
)
```

Arguments

`raw`	Numeric vector of raw scores
`group`	Numeric vector of grouping variable, e. g. grade. If no group or age variable is provided, conventional norming is applied
`age`	Numeric vector with chronological age, please additionally specify width of window
`width`	Size of the moving window in case an age vector is used
`weights`	Vector or variable name in the dataset with weights for each individual case. It can be used to compensate for moderate imbalances due to insufficient norm data stratification. Weights should be numerical and positive.

`scale`	type of norm scale, either T (default), IQ, z or percentile (= no transformation); a double vector with the mean and standard deviation can as well, be provided f. e. c(10, 3) for Wechsler scale index points
`method`	Ranking method in case of bindings, please provide an index, choosing from the following methods: 1 = Blom (1958), 2 = Tukey (1949), 3 = Van der Warden (1952), 4 = Rankit (default), 5 = Levenbach (1953), 6 = Filliben (1975), 7 = Yu & Huang (2001)
`descend`	ranking order (default descent = FALSE): inverses the ranking order with higher raw scores getting lower norm scores; relevant for example when norming error scores, where lower scores mean higher performance
`k`	The power constant. Higher values result in more detailed approximations but have the danger of over-fit (max = 6). If not set, it uses t and if both parameters are NULL, k is set to 5.
`t`	The age power parameter (max = 6). If not set, it uses k and if both parameters are NULL, k is set to 3, since age trajectories are most often well captured by cubic polynomials.
`terms`	Selection criterion for model building. The best fitting model with this number of terms is used
`R2`	Adjusted R square as a stopping criterion for the model building (default R2 = 0.99)

Value

cnorm object including the ranked raw data and the regression model

References

1. Gary, S. & Lenhard, W. (2021). In norming we trust. Diagnostica.
2. Gary, S., Lenhard, W. & Lenhard, A. (2021). Modelling Norm Scores with the cNORM Package in R. Psych, 3(3), 501-521. https://doi.org/10.3390/psych3030033
3. Lenhard, A., Lenhard, W., Suggate, S. & Segerer, R. (2016). A continuous solution to the norming problem. Assessment, Online first, 1-14. doi:10.1177/1073191116656437
4. Lenhard, A., Lenhard, W., Gary, S. (2018). Continuous Norming (cNORM). The Comprehensive R Network, Package cNORM, available: https://CRAN.R-project.org/package=cNORM
5. Lenhard, A., Lenhard, W., Gary, S. (2019). Continuous norming of psychometric tests: A simulation study of parametric and semi-parametric approaches. PLoS ONE, 14(9), e0222279. doi:10.1371/journal.pone.0222279
6. Lenhard, W., & Lenhard, A. (2020). Improvement of Norm Score Quality via Regression-Based Continuous Norming. Educational and Psychological Measurement(Online First), 1-33. https://doi.org/10.1177/0013164420928457

See Also

rankByGroup, rankBySlidingWindow, computePowers, bestModel

Examples

```
## Not run:
# Using this function with the example dataset 'elfe'

# Conventional norming (no modelling over age)
cnorm(raw=elfe$raw)

# Continuous norming
# You can use the 'getGroups()' function to set up grouping variable in case,
# you have a continuous age variable.
cnorm.elfe <- cnorm(raw = elfe$raw, group = elfe$group)

# return norm tables including 90% confidence intervals for a
# test with a reliability of r = .85; table are set to mean of quartal
# in grade 3 (children completed 2 years of schooling)
normTable(c(2.125, 2.375, 2.625, 2.875), cnorm.elfe, CI = .90, reliability = .95)

# ... or instead of raw scores for norm scores, the other way round
rawTable(c(2.125, 2.375, 2.625, 2.875), cnorm.elfe, CI = .90, reliability = .95)


# Using a continuous age variable instead of distinct groups, using a sliding
# window for percentile estimation. Please specify continuos variable for age
# and the sliding window size.
cnorm.ppvt.continuous <- cnorm(raw = ppvt$raw, age = ppvt$age, width=1)


# In case of unbalanced datasets, deviating from the census, the norm data
# can be weighted by the means of raking / post stratification. Please generate
# the weights with the computeWeights() function and pass them as the weights
# parameter. For computing the weights, please specify a data.frame with the
# population margins (further information is available in the computeWeights
# function). A demonstration based on sex and migration status in vocabulary
# development (ppvt dataset):
margins <- data.frame(variables = c("sex", "sex",
                                    "migration", "migration"),
                      levels = c(1, 2, 0, 1),
                      share = c(.52, .48, .7, .3))
weights <- computeWeights(ppvt, margins)
model <- cnorm(raw = ppvt$raw, group=ppvt$group, weights = weights)

## End(Not run)
```

---

cnorm.cv *Cross-validation for Term Selection in cNORM*

Description

Assists in determining the optimal number of terms for the regression model using repeated Monte Carlo cross-validation. It leverages an 80-20 split between training and validation data, with stratification by norm group or random sample in case of using sliding window ranking.

Usage

```
cnorm.cv(
  data,
  formula = NULL,
  repetitions = 5,
  norms = TRUE,
  min = 1,
  max = 12,
  cv = "full",
  pCutoff = NULL,
  width = NA,
  raw = NULL,
  group = NULL,
  age = NULL,
  weights = NULL
)
```

Arguments

`data`	Data frame of norm sample or a cnorm object. Should have ranking, powers, and interaction of L and A.
`formula`	Formula from an existing regression model; min/max functions ignored. If using a cnorm object, this is automatically fetched.
`repetitions`	Number of repetitions for cross-validation.
`norms`	If TRUE, computes norm score crossfit and R^2. Note: Computationally intensive.
`min`	Start with a minimum number of terms (default = 1).
`max`	Maximum terms in model, up to (k + 1) * (t + 1) - 1.
`cv`	"full" (default) splits data into training/validation, then ranks. Otherwise, expects a pre-ranked dataset.
`pCutoff`	Checks stratification for unbalanced data. Performs a t-test per group. Default set to 0.2 to minimize beta error.
`width`	If provided, ranking done via 'rankBySlidingWindow'. Otherwise, by group.
`raw`	Name of the raw score variable.
`group`	Name of the grouping variable.
`age`	Name of the age variable.
`weights`	Name of the weighting parameter.

Details

Successive models, with an increasing number of terms, are evaluated, and the RMSE for raw scores plotted. This encompasses the training, validation, and entire dataset. If 'norms' is set to TRUE (default), the function will also calculate the mean norm score reliability and crossfit measures. Note that due to the computational requirements of norm score calculations, execution can be slow, especially with numerous repetitions or terms.

When ‘cv‘ is set to "full" (default), both test and validation datasets are ranked separately, providing comprehensive cross-validation. For a more streamlined validation process focused only on modeling, a pre-ranked dataset can be used. The output comprises RMSE for raw score models, norm score R^2, delta R^2, crossfit, and the norm score SE according to Oosterhuis, van der Ark, & Sijtsma (2016).

For assessing overfitting:

$$CROSSFIT = R(Training; Model)^2 / R(Validation; Model)^2$$

A CROSSFIT > 1 suggests overfitting, < 1 suggests potential underfitting, and values around 1 are optimal, given a low raw score RMSE and high norm score validation R^2.

Suggestions for ideal model selection:

- Visual inspection of percentiles with ‘plotPercentiles‘ or ‘plotPercentileSeries‘.
- Pair visual inspection with repeated cross-validation (e.g., 10 repetitions).
- Aim for low raw score RMSE and high norm score R^2, avoiding terms with significant overfit (e.g., crossfit > 1.1).

Value

Table with results per term number: RMSE for raw scores, R^2 for norm scores, and crossfit measure.

References

Oosterhuis, H. E. M., van der Ark, L. A., & Sijtsma, K. (2016). Sample Size Requirements for Traditional and Regression-Based Norms. Assessment, 23(2), 191–202. https://doi.org/10.1177/1073191115580638

See Also

Other model: `bestModel()`, `checkConsistency()`, `derive()`, `modelSummary()`, `print.cnorm()`, `printSubset()`, `rangeCheck()`, `regressionFunction()`, `summary.cnorm()`

Examples

```
## Not run:
# Example: Plot cross-validation RMSE by number of terms (up to 9) with three repetitions.
result <- cnorm(raw = elfe$raw, group = elfe$group)
cnorm.cv(result$data, min = 2, max = 9, repetitions = 3)

# Using a cnorm object examines the predefined formula.
cnorm.cv(result, repetitions = 1)

# For cross-validation without a cnorm model, rank data first and compute powers:
data <- rankByGroup(data = elfe, raw = "raw", group = "group")
data <- computePowers(data)
cnorm.cv(data)

# Specify formulas deliberately:
data <- rankByGroup(data = elfe, raw = "raw", group = "group")
```

```
data <- computePowers(data)
cnorm.cv(data, formula = formula(raw ~ L3 + L1A1 + L3A3 + L4 + L5))

## End(Not run)
```

`cNORM.GUI` *Launcher for the graphical user interface of cNORM*

Description

Launcher for the graphical user interface of cNORM

Usage

```
cNORM.GUI(launch.browser = TRUE)
```

Arguments

`launch.browser` Default TRUE; automatically open browser for GUI

Examples

```
## Not run:
# Launch graphical user interface
cNORM.GUI()

## End(Not run)
```

`computePowers` *Compute powers of the explanatory variable a as well as of the person location l (data preparation)*

Description

The function computes powers of the norm variable e. g. T scores (location, L), an explanatory variable, e. g. age or grade of a data frame (age, A) and the interactions of both (L X A). The k variable indicates the degree up to which powers and interactions are build. These predictors can be used later on in the `bestModel` function to model the norm sample. Higher values of k allow for modeling the norm sample closer, but might lead to over-fit. In general k = 3 or k = 4 (default) is sufficient to model human performance data. For example, k = 2 results in the variables L1, L2, A1, A2, and their interactions L1A1, L2A1, L1A2 and L2A2 (but k = 2 is usually not sufficient for the modeling). Please note, that you do not need to use a normal rank transformed scale like T r IQ, but you can as well use the percentiles for the 'normValue' as well.

Usage

```
computePowers(
  data,
  k = 5,
  norm = NULL,
  age = NULL,
  t = 3,
  covariate = NULL,
  silent = FALSE
)
```

Arguments

`data`	data.frame with the norm data
`k`	degree
`norm`	the variable containing the norm data in the data.frame; might be T scores, IQ scores, percentiles ...
`age`	Explanatory variable like age or grade, which was as well used for the grouping. Can be either the grouping variable itself or a finer grained variable like the exact age. Other explanatory variables can be used here instead an age variable as well, as long as the variable is at least ordered metric, e. g. language or development levels ... The label 'age' is used, as this is the most common field of application.
`t`	the age power parameter (default NULL). If not set, cNORM automatically uses k. The age power parameter can be used to specify the k to produce rectangular matrices and specify the course of scores per independently from k
`covariate`	Include a binary covariate into the preparation and subsequently modeling, either by specifying the variable name or including the variable itself. If this has already been done in the ranking, the function uses the according variable. BEWARE! Not all subsequent functions are already prepared for it. It is an experimental feature and may lead to unstable models subsequently.
`silent`	set to TRUE to suppress messages

Value

data.frame with the powers and interactions of location and explanatory variable / age

See Also

bestModel

Other prepare: `prepareData()`, `rankByGroup()`, `rankBySlidingWindow()`

Examples

```
# Dataset with grade levels as grouping
data.elfe <- rankByGroup(elfe)
data.elfe <- computePowers(data.elfe)
```

```
# Dataset with continuous age variable and k = 5
data.ppvt <- rankByGroup(ppvt)
data.ppvt <- computePowers(data.ppvt, age = "age", k = 5)
```

`computeWeights` *Weighting of cases through iterative proportional fitting (Raking)*

Description

Computes and standardizes weights via raking to compensate for non-stratified samples. It is based on the implementation in the survey R package. It reduces data collection #' biases in the norm data by the means of post stratification, thus reducing the effect of unbalanced data in percentile estimation and norm data modeling.

Usage

```
computeWeights(data, population.margins, standardized = TRUE)
```

Arguments

`data`	data.frame with norm sample data.
`population.margins`	A data.frame including three columns, specifying the variable name in the original dataset used for data stratification, the factor level of the variable and the according population share. Please ensure, the original data does not include factor levels, not present in the population.margins. Additionally, summing up the shares of the different levels of a variable should result in a value near 1.0. The first column must specify the name of the stratification variable, the second the level and the third the proportion
`standardized`	If TRUE (default), the raking weights are scaled to weights/min(weights)

Details

This function computes standardized raking weights to overcome biases in norm samples. It generates weights, by drawing on the information of population shares (e. g. for sex, ethnic group, region ...) and subsequently reduces the influence of over-represented groups or increases under-represented cases. The returned weights are either raw or standardized and scaled to be larger than 0.

Raking in general has a number of advantages over post stratification and it additionally allows cNORM to draw on larger datasets, since less cases have to be removed during stratification. To use this function, additionally to the data, a data frame with stratification variables has to be specified. The data frame should include a row with (a) the variable name, (b) the level of the variable and (c) the according population proportion.

Value

a vector with the standardized weights

Examples

```
# cNORM features a dataset on vocabulary development (ppvt)
# that includes variables like sex or migration. In order
# to weight the data, we have to specify the population shares.
# According to census, the population includes 52% boys
# (factor level 1 in the ppvt dataset) and 70% / 30% of persons
# without / with a a history of migration (= 0 / 1 in the dataset).
# First we set up the popolation margins with all shares of the
# different levels:

margins <- data.frame(variables = c("sex", "sex",
                                    "migration", "migration"),
                      levels = c(1, 2, 0, 1),
                      share = c(.52, .48, .7, .3))
head(margins)

# Now we use the population margins to generate weights
# through raking

weights <- computeWeights(ppvt, margins)


# There are as many different weights as combinations of
# factor levels, thus only four in this specific case

unique(weights)


# To include the weights in the cNORM modelling, we have
# to pass them as weights. They are then used to set up
# weighted quantiles and as weights in the regession.

model <- cnorm(raw = ppvt$raw,
               group=ppvt$group,
               weights = weights)
```

`derivationTable` *Create a table based on first order derivative of the regression model for specific age*

Description

In order to check model assumptions, a table of the first order derivative of the model coefficients is created.

Usage

```
derivationTable(
  A,
```

```
  model,
  minNorm = NULL,
  maxNorm = NULL,
  step = 0.1,
  covariate = NULL
)
```

Arguments

A	the age
model	The regression model or a cnorm object
minNorm	The lower bound of the norm value range
maxNorm	The upper bound of the norm value range
step	Stepping parameter with lower values indicating higher precision
covariate	In case, a covariate has been used, please specify the degree of the covariate / the specific value here.

Value

data.frame with norm scores and the predicted scores based on the derived regression function

See Also

plotDerivative, derive

Other predict: getNormCurve(), normTable(), predictNorm(), predictRaw(), rawTable()

Examples

```
# Generate cnorm object from example data
cnorm.elfe <- cnorm(raw = elfe$raw, group = elfe$group)

# retrieve function for time point 6
d <- derivationTable(6, cnorm.elfe, step = 0.5)
```

---

derive *Derivative of regression model*

Description

Calculates the derivative of the location / norm value from the regression model with the first derivative as the default. This is useful for finding violations of model assumptions and problematic distribution features as f. e. bottom and ceiling effects, non-progressive norm scores within an age group or in general #' intersecting percentile curves.

Usage

```
derive(model, order = 1, covariate = NULL)
```

Arguments

model	The regression model or a cnorm object
order	The degree of the derivate, default: 1
covariate	In case, a covariate has been used, please specify the degree of the covariate / the specific value here.

Value

The derived coefficients

See Also

Other model: bestModel(), checkConsistency(), cnorm.cv(), modelSummary(), print.cnorm(), printSubset(), rangeCheck(), regressionFunction(), summary.cnorm()

Examples

```
normData <- prepareData(elfe)
m <- bestModel(normData)
derivedCoefficients <- derive(m)
```

---

elfe *Sentence completion test from ELFE 1-6*

Description

A dataset containing the raw data of 1400 students from grade 2 to 5 in the sentence comprehension test from ELFE 1-6 (Lenhard & Schneider, 2006). In this test, students are presented lists of sentences with one gap. The student has to fill in the correct solution by selecting from a list of 5 alternatives per sentence. The alternatives include verbs, adjectives, nouns, pronouns and conjunctives. Each item stems from the same word type. The text is speeded, with a time cutoff of 180 seconds. The variables are as follows:

Usage

```
elfe
```

Format

A data frame with 1400 rows and 3 variables:

personID ID of the student

group grade level, with x.5 indicating the end of the school year and x.0 indicating the middle of the school year

raw the raw score of the student, spanning values from 0 to 28

A data frame with 1400 rows and 3 columns

Source

https://www.psychometrica.de/elfe2.html

References

Lenhard, W. & Schneider, W.(2006). Ein Leseverstaendnistest fuer Erst- bis Sechstklaesser. Goettingen/Germany: Hogrefe.

Examples

```
# prepare data, retrieve model and plot percentiles
data.elfe <- prepareData(elfe)
model.elfe <- bestModel(data.elfe)
plotPercentiles(data.elfe, model.elfe)
```

---

epm *Simulated dataset (Educational and Psychological Measurement, EPM)*

Description

A simulated dataset, based on the the simRasch function. The data were generated on the basis of a 1PL IRT model with 50 items with a normal distribution and a mean difficulty of m = 0 and sd = 1 and 1400 cases. The age trajectory features a curve linear increase wit a slight scissor effect. The sample consists of seven age groups with 200 cases each and it includes information on the latent ability, the age specific latent ability and norm scores based on conventional norming with differing granularity of the age brackets.

Usage

```
epm
```

Format

A data frame with 1400 rows and 10 variables:

raw the raw score

ageSpecificZ the age specific latent ability, z standardized

latentTrait the overall latent trait with respect to the population model

age the chronological age

halfYearGroup grouping variable based on six month age brackets

spcnT Resulting norm score of cNORM, based on the automatic model selection

T1 conventional T scores on the basis of one month age brackets

T3 conventional T scores on the basis of three month age brackets

T6 conventional T scores on the basis of six month age brackets

T12 conventional T scores on the basis of one year age brackets

A data frame with 1400 rows and 10 columns

Source

https://osf.io/ntydc/

References

Lenhard, W. & Lenhard, A. (2020). Improvement of Norm Score Quality via Regression-Based Continuous Norming. Educational and Psychological Measurement. https://doi.org/10.1177/0013164420928457

Examples

```
## Not run:
# Example with continuous age variable
data.epm <- prepareData(epm, raw=epm$raw, group=epm$halfYearGroup, age=epm$age)
model.epm <- bestModel(data.epm)

## End(Not run)
```

---

getGroups *Determine groups and group means*

Description

Helps to split the continuous explanatory variable into groups and assigns the group mean. The groups can be split either into groups of equal size (default) or equal number of observations.

Usage

```
getGroups(x, n = NULL, equidistant = FALSE)
```

Arguments

`x`	The continuous variable to be split
`n`	The number of groups; if NULL then the function determines a number of groups with usually 100 cases or 3 <= n <= 20.
`equidistant`	If set to TRUE, builds equidistant interval, otherwise (default) with equal number of observations

Value

vector with group means for each observation

Examples

```
x <- rnorm(1000, m = 50, sd = 10)
m <- getGroups(x, n = 10)
```

`getNormCurve` — *Computes the curve for a specific T value*

Description

As with this continuous norming regression approach, raw scores are modeled as a function of age and norm score (location), getNormCurve is a straightforward approach to show the raw score development over age, while keeping the norm value constant. This way, e. g. academic performance or intelligence development of a specific ability is shown.

Usage

```
getNormCurve(
  norm,
  model,
  minAge = NULL,
  maxAge = NULL,
  step = 0.1,
  minRaw = NULL,
  maxRaw = NULL,
  covariate = NULL
)
```

Arguments

`norm`	The specific norm score, e. g. T value
`model`	The model from the regression modeling or a cnorm object
`minAge`	Age to start from
`maxAge`	Age to stop at

`step`	Stepping parameter for the precision when retrieving of the values, lower values indicate higher precision (default 0.1).
`minRaw`	lower bound of the range of raw scores (default = 0)
`maxRaw`	upper bound of raw scores
`covariate`	In case, a covariate has been used, please specify the degree of the covariate or the specific value here.

Value

data.frame of the variables raw, age and norm

See Also

Other predict: `derivationTable()`, `normTable()`, `predictNorm()`, `predictRaw()`, `rawTable()`

Examples

```
# Generate cnorm object from example data
cnorm.elfe <- cnorm(raw = elfe$raw, group = elfe$group)
getNormCurve(35, cnorm.elfe)
```

`getNormScoreSE` *Calculates the standard error (SE) or root mean square error (RMSE) of the norm scores In case of large datasets, both results should be almost identical*

Description

Calculates the standard error (SE) or root mean square error (RMSE) of the norm scores In case of large datasets, both results should be almost identical

Usage

```
getNormScoreSE(model, type = 2)
```

Arguments

`model`	a cnorm object
`type`	either '1' for the standard error senso Oosterhuis et al. (2016) or '2' for the RMSE (default)

Value

The standard error (SE) of the norm scores sensu Oosterhuis et al. (2016) or the RMSE

References

Oosterhuis, H. E. M., van der Ark, L. A., & Sijtsma, K. (2016). Sample Size Requirements for Traditional and Regression-Based Norms. Assessment, 23(2), 191–202. https://doi.org/10.1177/1073191115580638

life *Life expectancy at birth from 1960 to 2017*

Description

The data is available by the courtesy of the World Bank under Creative Commons Attribution 4.0 (CC-BY 4.0). It includes the life expectancy at birth on nation level from 1960 to 2017. The data has been converted to long data format, aggregates for groups of nations and missings have been deleted and a grouping variable with a broader scope spanning 4 years each has been added. It shows, that it can be better to reduce predictors. The model does not converge anymore after using 8 predictors and the optimal solution is achieved with four predictors, equaling R2=.9825.

Usage

```
life
```

Format

A data frame with 11182 rows and 4 variables:

Country The name of the country

year reference year of data collection

life the life expectancy at birth

group a grouping variable based on 'year' but with a lower resolution; spans intervals of 4 years each

A data frame with 11182 rows and 4 columns

Source

https://data.worldbank.org/indicator/sp.dyn.le00.in

References

The World Bank (2018). Life expectancy at birth, total (years). Data Source World Development Indicators available https://data.worldbank.org/indicator/sp.dyn.le00.in (date of retrieval: 01/09/2018)

Examples

```
## Not run:
# data preparation
data.life <- rankByGroup(life, raw="life")
data.life <- computePowers(data.life, age="year")

#determining best suiting model by plotting series
model.life <- bestModel(data.life, raw="life")
plotPercentileSeries(data.life, model.life, end=10)
```

```
# model with four predictors seems to work best
model2.life <- bestModel(data.life, raw="life", terms=4)

## End(Not run)
```

`modelSummary` *Prints the results and regression function of a cnorm model*

Description

Prints the results and regression function of a cnorm model

Usage

```
modelSummary(object, ...)
```

Arguments

`object`	A regression model or cnorm object
`...`	additional parameters

Value

A report on the regression function, weights, R2 and RMSE

See Also

Other model: `bestModel()`, `checkConsistency()`, `cnorm.cv()`, `derive()`, `print.cnorm()`, `printSubset()`, `rangeCheck()`, `regressionFunction()`, `summary.cnorm()`

`mortality` *Mortality of infants per 1000 life birth from 1960 to 2017*

Description

The data is available by the courtesy of the World Bank under Creative Commons Attribution 4.0 (CC-BY 4.0). It includes the mortality rate of life birth per country from 1960 to 2017. The data has been converted to long data format, aggregates for groups of nations and missings have been deleted and a grouping variable with a broader scope spanning 4 years each has been added. It can be used for demonstrating intersecting percentile curves at bottom effects.

Usage

```
mortality
```

Format

A data frame with 9547 rows and 4 variables:

Country The name of the country

year reference year of data collection

mortality the mortality per 1000 life born children

group grouping variable based on 'year' with a lower resolution; spans intervals of 4 years each

Source

https://data.worldbank.org/indicator/SP.DYN.IMRT.IN

References

The World Bank (2018). Mortality rate, infant (per 1,000 live births). Data Source available https://data.worldbank.org/indicator/SP.DYN.IMRT.IN (date of retrieval: 02/09/2018)

Examples

```
# data preparation
data.mortality <- rankByGroup(mortality, raw="mortality")
data.mortality <- computePowers(data.mortality, age="year")

# modeling
model.mortality <- bestModel(data.mortality, raw="mortality")
plotSubset(model.mortality, type = 0)
plotPercentileSeries(data.mortality, model.mortality, end=9, percentiles = c(.1, .25, .5, .75, .9))
```

`normTable` *Create a norm table based on model for specific age*

Description

This function generates a norm table for a specific age based on the regression model by assigning raw scores to norm scores. Please specify the range of norm scores, you want to cover. A T value of 25 corresponds to a percentile of .6. As a consequence, specifying a range of T = 25 to T = 75 would cover 98.4 the population. Please be careful when extrapolating vertically (at the lower and upper end of the age specific distribution). Depending on the size of your standardization sample, extreme values with T < 20 or T > 80 might lead to inconsistent results. In case a confidence coefficient (CI, default .9) and the reliability is specified, confidence intervals are computed for the true score estimates, including a correction for regression to the mean (Eid & Schmidt, 2012, p. 272).

Usage

```
normTable(
  A,
  model,
  minNorm = NULL,
  maxNorm = NULL,
  minRaw = NULL,
  maxRaw = NULL,
  step = NULL,
  covariate = NULL,
  monotonuous = TRUE,
  CI = 0.9,
  reliability = NULL,
  pretty = T
)
```

Arguments

A	the age as single value or a vector of age values
model	The regression model or a cnorm object
minNorm	The lower bound of the norm score range
maxNorm	The upper bound of the norm score range
minRaw	clipping parameter for the lower bound of raw scores
maxRaw	clipping parameter for the upper bound of raw scores
step	Stepping parameter with lower values indicating higher precision
covariate	In case, a covariate has been used, please specify the degree of the covariate / the specific value here.
monotonuous	corrects for decreasing norm scores in case of model inconsistencies (default)
CI	confidence coefficient, ranging from 0 to 1, default .9
reliability	coefficient, ranging between 0 to 1
pretty	Format table by collapsing intervals and rounding to meaningful precision

Value

either data.frame with norm scores, predicted raw scores and percentiles in case of simple A value or a list #' of norm tables if vector of A values was provided

References

Eid, M. & Schmidt, K. (2012). Testtheorie und Testkonstruktion. Hogrefe.

See Also

rawTable

Other predict: derivationTable(), getNormCurve(), predictNorm(), predictRaw(), rawTable()

Examples

```
# Generate cnorm object from example data
cnorm.elfe <- cnorm(raw = elfe$raw, group = elfe$group)

# create single norm table
norms <- normTable(3.5, cnorm.elfe, minNorm = 25, maxNorm = 75, step = 0.5)

# create list of norm tables
norms <- normTable(c(2.5, 3.5, 4.5), cnorm.elfe,
  minNorm = 25, maxNorm = 75,
  step = 1, minRaw = 0, maxRaw = 26
)
```

`plot.cnorm` *S3 function for plotting cnorm objects*

Description

S3 function for plotting cnorm objects

Usage

```
## S3 method for class 'cnorm'
plot(x, y, ...)
```

Arguments

x	the cnorm object
y	the type of plot as a string, can be one of 'raw' (1), 'norm' (2), 'curves' (3), 'percentiles' (4), 'series' (5), 'subset' (6), or 'derivative' (7), either as a string or the according index
...	additional parameters for the specific plotting function

See Also

Other plot: `plotDensity()`, `plotDerivative()`, `plotNormCurves()`, `plotNorm()`, `plotPercentileSeries()`, `plotPercentiles()`, `plotRaw()`, `plotSubset()`

`plotCnorm` *General convencience plotting function*

Description

General convencience plotting function

Usage

```
plotCnorm(x, y, ...)
```

Arguments

`x`	a cnorm object
`y`	the type of plot as a string, can be one of 'raw' (1), 'norm' (2), 'curves' (3), 'percentiles' (4), 'series' (5), 'subset' (6), or 'derivative' (7), either as a string or the according index
`...`	additional parameters for the specific plotting function

`plotDensity` *Plot the density function per group by raw score*

Description

The function plots the density curves based on the regression model against the actual percentiles from the raw data. As in 'plotNormCurves', please check for inconsistent curves, especially curves showing implausible shapes as f. e. violations of biuniqueness.

Usage

```
plotDensity(
  model,
  minRaw = NULL,
  maxRaw = NULL,
  minNorm = NULL,
  maxNorm = NULL,
  group = NULL,
  covariate = NULL
)
```

Arguments

`model`	The model from the bestModel function or a cnorm object
`minRaw`	Lower bound of the raw score
`maxRaw`	Upper bound of the raw score
`minNorm`	Lower bound of the norm score
`maxNorm`	Upper bound of the norm score
`group`	Column of groups to plot
`covariate`	In case, a covariate has been used, please specify the degree of the covariate / the specific value here.

See Also

plotNormCurves, plotPercentiles

Other plot: `plot.cnorm()`, `plotDerivative()`, `plotNormCurves()`, `plotNorm()`, `plotPercentileSeries()`, `plotPercentiles()`, `plotRaw()`, `plotSubset()`

Examples

```
# Load example data set, compute model and plot results for age values 2, 4 and 6
result <- cnorm(raw = elfe$raw, group = elfe$group)
plotDensity(result, group = c (2, 4, 6))
```

`plotDerivative` *Plot first order derivative of regression model*

Description

Plots the scores obtained via the first order derivative of the regression model in dependence of the norm score. The results indicate the progression of the norm scores within each age group. The regression based modeling approach relies on the assumption of a linear progression of the norm scores. Negative scores in the first order derivative indicate a violation of this assumption. Scores near zero are typical for bottom and ceiling effects in the raw data. The regression models usually converge within the range of the original values. In case of vertical and horizontal extrapolation, with increasing distance to the original data, the risk of assumption violation increases as well. ATTENTION: plotDerivative is currently still incompatible with reversed raw score scales ('descent' option)

Usage

```
plotDerivative(
  model,
  minAge = NULL,
  maxAge = NULL,
  minNorm = NULL,
  maxNorm = NULL,
```

```
  stepAge = 0.2,
  stepNorm = 1,
  order = 1
)
```

Arguments

model	The model from the bestModel function or a cnorm object
minAge	Age to start with checking
maxAge	Upper end of the age check
minNorm	Lower end of the norm score range, in case of T scores, 25 might be good
maxNorm	Upper end of the norm score range, in case of T scores, 25 might be good
stepAge	Stepping parameter for the age check, usually 1 or 0.1; lower values indicate higher precision / closer checks
stepNorm	Stepping parameter for norm scores
order	Degree of the derivative (default = 1)

See Also

checkConsistency, bestModel, derive

Other plot: plot.cnorm(), plotDensity(), plotNormCurves(), plotNorm(), plotPercentileSeries(), plotPercentiles(), plotRaw(), plotSubset()

Examples

```
# Load example data set, compute model and plot results
result <- cnorm(raw = elfe$raw, group = elfe$group)
plotDerivative(result, minAge=2, maxAge=5, step=.2, minNorm=25, maxNorm=75, stepNorm=1)
```

---

plotNorm *Plot manifest and fitted norm scores*

Description

The function plots the manifest norm score against the fitted norm score from the inverse regression model per group. This helps to inspect the precision of the modeling process. The scores should not deviate too far from regression line.

Usage

```
plotNorm(data, model, group = "", minNorm = NULL, maxNorm = NULL, type = 0)
```

Arguments

data	The raw data within a data.frame or a cnorm object
model	The regression model (optional)
group	The grouping variable, use empty string for no group
minNorm	lower bound of fitted norm scores
maxNorm	upper bound of fitted norm scores
type	Type of display: 0 = plot manifest against fitted values, 1 = plot manifest against difference values

See Also

Other plot: plot.cnorm(), plotDensity(), plotDerivative(), plotNormCurves(), plotPercentileSeries(), plotPercentiles(), plotRaw(), plotSubset()

Examples

```
# Load example data set, compute model and plot results
## Not run:
result <- cnorm(raw = elfe$raw, group = elfe$group)
plotNorm(result, group="group", minNorm=25, maxNorm=75)

## End(Not run)
```

---

plotNormCurves *Plot norm curves*

Description

The function plots the norm curves based on the regression model. Please check the function for inconsistent curves: The different curves should not intersect. Violations of this assumption are a strong indication for violations of model assumptions in modeling the relationship between raw and norm scores. There are several reasons, why this might occur:

1. Vertical extrapolation: Choosing extreme norm scores, e. g. scores -3 <= x and x >= 3 In order to model these extreme scores, a large sample dataset is necessary.
2. Horizontal extrapolation: Taylor polynomials converge in a certain radius. Using the model scores outside the original dataset may lead to inconsistent results.
3. The data cannot be modeled with Taylor polynomials, or you need another power parameter (k) or R2 for the model.

In general, extrapolation (point 1 and 2) can carefully be done to a certain degree outside the original sample, but it should in general be handled with caution. checkConsistency and derivationPlot can be used to further inspect the model.

Usage

```
plotNormCurves(
  model,
  normList = NULL,
  minAge = NULL,
  maxAge = NULL,
  step = 0.1,
  minRaw = NULL,
  maxRaw = NULL,
  covariate = NULL
)
```

Arguments

`model`	The model from the bestModel function or a cnorm object
`normList`	Vector with norm scores to display
`minAge`	Age to start with checking
`maxAge`	Upper end of the age check
`step`	Stepping parameter for the age check, usually 1 or 0.1; lower scores indicate higher precision / closer checks
`minRaw`	Lower end of the raw score range, used for clipping implausible results (default = 0)
`maxRaw`	Upper end of the raw score range, used for clipping implausible results
`covariate`	In case, a covariate has been used, please specify the degree of the covariate / the specific value here.

See Also

checkConsistency, derivationPlot, plotPercentiles

Other plot: `plot.cnorm()`, `plotDensity()`, `plotDerivative()`, `plotNorm()`, `plotPercentileSeries()`, `plotPercentiles()`, `plotRaw()`, `plotSubset()`

Examples

```
# Load example data set, compute model and plot results
normData <- prepareData(elfe)
m <- bestModel(data = normData)
plotNormCurves(m, minAge=2, maxAge=5)
```

plotPercentiles *Plot norm curves against actual percentiles*

Description

The function plots the norm curves based on the regression model against the actual percentiles from the raw data. As in 'plotNormCurves', please check for inconsistent curves, especially intersections. Violations of this assumption are a strong indication for problems in modeling the relationship between raw and norm scores. In general, extrapolation (point 1 and 2) can carefully be done to a certain degree outside the original sample, but it should in general be handled with caution. The original percentiles are displayed as distinct points in the according color, the model based projection of percentiles are drawn as lines. Please note, that the estimation of the percentiles of the raw data is done with the quantile function with the default settings. Please consult help(quantile) and change the 'type' parameter accordingly. In case, you get 'jagged' or disorganized percentile curve, try to reduce the 'k' parameter in modeling.

Usage

```
plotPercentiles(
  data,
  model,
  minRaw = NULL,
  maxRaw = NULL,
  minAge = NULL,
  maxAge = NULL,
  raw = NULL,
  group = NULL,
  percentiles = c(0.025, 0.1, 0.25, 0.5, 0.75, 0.9, 0.975),
  scale = NULL,
  type = 7,
  title = NULL,
  covariate = NULL
)
```

Arguments

data	The raw data including the percentiles and norm scores or a cnorm object
model	The model from the bestModel function (optional)
minRaw	Lower bound of the raw score (default = 0)
maxRaw	Upper bound of the raw score
minAge	Variable to restrict the lower bound of the plot to a specific age
maxAge	Variable to restrict the upper bound of the plot to a specific age
raw	The name of the raw variable
group	The name of the grouping variable; the distinct groups are automatically determined

percentiles	Vector with percentile scores, ranging from 0 to 1 (exclusive)
scale	The norm scale, either 'T', 'IQ', 'z', 'percentile' or self defined with a double vector with the mean and standard deviation, f. e. c(10, 3) for Wechsler scale index points; if NULL, scale information from the data preparation is used (default)
type	The type parameter of the quantile function to estimate the percentiles of the raw data (default 7)
title	custom title for plot
covariate	In case, a covariate has been used, please specify the degree of the covariate / the specific value here. If no covariate is specified, both degrees will be plotted.

See Also

plotNormCurves, plotPercentileSeries

Other plot: plot.cnorm(), plotDensity(), plotDerivative(), plotNormCurves(), plotNorm(), plotPercentileSeries(), plotRaw(), plotSubset()

Examples

```
# Load example data set, compute model and plot results
result <- cnorm(raw = elfe$raw, group = elfe$group)
plotPercentiles(result)
```

---

plotPercentileSeries *Generates a series of plots with number curves by percentile for different models*

Description

This functions makes use of 'plotPercentiles' to generate a series of plots with different number of predictors. It draws on the information provided by the model object to determine the bounds of the modeling (age and standard score range). It can be used as an additional model check to determine the best fitting model. Please have a look at the ' plotPercentiles' function for further information.

Usage

```
plotPercentileSeries(
  data,
  model,
  start = 1,
  end = NULL,
  group = NULL,
  percentiles = c(0.025, 0.1, 0.25, 0.5, 0.75, 0.9, 0.975),
  type = 7,
  filename = NULL
)
```

Arguments

`data`	The raw data including the percentiles and norm scores or a cnorm object
`model`	The model from the bestModel function (optional)
`start`	Number of predictors to start with
`end`	Number of predictors to end with
`group`	The name of the grouping variable; the distinct groups are automatically determined
`percentiles`	Vector with percentile scores, ranging from 0 to 1 (exclusive)
`type`	The type parameter of the quantile function to estimate the percentiles of the raw data (default 7)
`filename`	Prefix of the filename. If specified, the plots are saves as png files in the directory of the workspace, instead of displaying them

Value

the complete list of plots

See Also

plotPercentiles

Other plot: `plot.cnorm()`, `plotDensity()`, `plotDerivative()`, `plotNormCurves()`, `plotNorm()`, `plotPercentiles()`, `plotRaw()`, `plotSubset()`

Examples

```
# Load example data set, compute model and plot results
result <- cnorm(raw = elfe$raw, group = elfe$group)
plotPercentileSeries(result, start=1, end=5, group="group")
```

plotRaw *Plot manifest and fitted raw scores*

Description

The function plots the raw data against the fitted scores from the regression model per group. This helps to inspect the precision of the modeling process. The scores should not deviate too far from regression line.

Usage

```
plotRaw(data, model, group = NULL, raw = NULL, type = 0)
```

Arguments

`data`	The raw data within a data.frame or cnorm object
`model`	The regression model (optional)
`group`	The grouping variable
`raw`	The raw score variable
`type`	Type of display: 0 = plot manifest against fitted values, 1 = plot manifest against difference values

See Also

Other plot: `plot.cnorm()`, `plotDensity()`, `plotDerivative()`, `plotNormCurves()`, `plotNorm()`, `plotPercentileSeries()`, `plotPercentiles()`, `plotSubset()`

Examples

```
# Compute model with example dataset and plot results
result <- cnorm(raw = elfe$raw, group = elfe$group)
plotRaw(result)
```

---

plotSubset — *Evaluate information criteria for regression model*

Description

Plots the information criterion - either Cp (default) or BIC - against the adjusted R square of the feature selection in the modeling process. Both BIC and Mallow's Cp are measures to avoid over-fitting. Please choose the model that has a high information criterion, while modeling the original data as close as possible. R2 adjusted values of ~ .99 might work well, depending on your scenario. In other words: Look out for the elbow in the curve and choose th model where the information criterion begins to drop. Nonetheless, inspect the according model with `plotPercentiles(data, group)` to visually inspect the course of the percentiles. In the plot, Mallow's Cp is log transformed and the BIC is always highly negative. The R2 cutoff that was specified in the bestModel function is displayed as a dashed line.

Usage

```
plotSubset(model, type = 0, index = FALSE)
```

Arguments

`model`	The regression model from the bestModel function or a cnorm object
`type`	Type of chart with 0 = adjusted R2 by number of predictors, 1 = log transformed Mallow's Cp by adjusted R2, 2 = Bayesian Information Criterion (BIC) by adjusted R2, 3 = Root Mean Square Error (RMSE), 4 = Residual Sum of Squares by number, 5 = F-test statistic for consecutive models and 6 = p-value for model tests of predictors
`index`	add index labels to data points

See Also

bestModel, plotPercentiles, printSubset

Other plot: plot.cnorm(), plotDensity(), plotDerivative(), plotNormCurves(), plotNorm(), plotPercentileSeries(), plotPercentiles(), plotRaw()

Examples

```
# Compute model with example data and plot information function
cnorm.model <- cnorm(raw = elfe$raw, group = elfe$group)
plotSubset(cnorm.model)
```

---

ppvt *Vocabulary development from 2.5 to 17*

Description

A dataset based on an unstratified sample of PPVT4 data (German adaption). The PPVT4 consists of blocks of items with 12 items each. Each item consists of 4 pictures. The test taker is given a word orally and he or she has to point out the picture matching the oral word. Bottom and ceiling blocks of items are determined according to age and performance. For instance, when a student knows less than 4 word from a block of 12 items, the testing stops. The sample is not identical with the norm sample and includes doublets of cases in order to align the sample size per age group. It is primarily intended for running the cNORM analyses with regard to modeling and stratification.

Usage

```
ppvt
```

Format

A data frame with 4542 rows and 6 variables:

age the chronological age of the child

sex the sex of the test taker, 1=male, 2=female

migration migration status of the family, 0=no, 1=yes

region factor specifiying the region, the data were collected; grouped into south, north, east and west

raw the raw score of the student, spanning values from 0 to 228

group age group of the child, determined by the getGroups()-function with 12 equidistant age groups

A data frame with 5600 rows and 9 columns

Source

https://www.psychometrica.de/ppvt4.html

References

Lenhard, A., Lenhard, W., Segerer, R. & Suggate, S. (2015). Peabody Picture Vocabulary Test - Revision IV (Deutsche Adaption). Frankfurt a. M./Germany: Pearson Assessment.

Examples

```
## Not run:
# Example with continuous age variable, ranked with sliding window
model.ppvt.sliding <- cnorm(age=ppvt$age, raw=ppvt$raw, width=1)

# Example with age groups; you might first want to experiment with
# the granularity of the groups via the 'getGroups()' function
model.ppvt.group <- cnorm(group=ppvt$group, raw=ppvt$raw) # with predefined groups
model.ppvt.group <- cnorm(group=getGroups(ppvt$age, n=15, equidistant = T),
                          raw=ppvt$raw) # groups built 'on the fly'


# plot information function
plot(model.ppvt.group, "subset")

# check model consistency
checkConsistency(model.ppvt.group)

# plot percentiles
plot(model.ppvt.group, "percentiles")

## End(Not run)
```

---

predictBeta — *Predicts beta coefficients in dependence of age*

Description

Predicts beta coefficients in dependence of age

Usage

```
predictBeta(model, x, group)
```

Arguments

`model`	An ‘betaContinuous‘ model output
`x`	A vector specifying the raw scores
`group`	A vector specifying the group variables for each raw score

Value

A data.frame with z scores and percentiles as well as predicted a and b values for the specific group

Examples

```
# Determies beta parameters and models these continuously
param <- betaByGroup(elfe$raw, elfe$group, 26)
beta.model <- betaContinuous(param, 4, 4)

# Calculates z scores
x <- c(15, 8, 11, 18)
newGroup <- c(3.9, 1.2, 4.5, 6.3)

predictBeta(beta.model, x, newGroup)
```

---

predictNorm — *Retrieve norm value for raw score at a specific age*

Description

This functions numericaly determines the norm score for raw scores depending on the level of the explanatory variable A, e. g. norm scores for raw scores at given ages.

Usage

```
predictNorm(
  raw,
  A,
  model,
  minNorm = NULL,
  maxNorm = NULL,
  force = FALSE,
  covariate = NULL,
  silent = FALSE
)
```

Arguments

raw	The raw value, either single numeric or numeric vector
A	the explanatory variable (e. g. age), either single numeric or numeric vector
model	The regression model or a cnorm object
minNorm	The lower bound of the norm score range
maxNorm	The upper bound of the norm score range
force	Try to resolve missing norm scores in case of inconsistent models
covariate	In case, a covariate has been used, please specify the degree of the covariate / the specific value here.
silent	set to TRUE to suppress messages

Value

The predicted norm score for a raw score, either single value or vector

See Also

Other predict: derivationTable(), getNormCurve(), normTable(), predictRaw(), rawTable()

Examples

```
# Generate cnorm object from example data
cnorm.elfe <- cnorm(raw = elfe$raw, group = elfe$group)

# return norm value for raw value 21 for grade 2, month 9
specificNormValue <- predictNorm(raw = 21, A = 2.75, cnorm.elfe)

# predicted norm scores for the elfe dataset
# predictNorm(elfe$raw, elfe$group, cnorm.elfe)
```

---

predictRaw *Predict single raw value*

Description

Most elementary function to predict raw score based on Location (L, T score), Age (grouping variable) and the coefficients from a regression model. WARNING! This function, and all functions depending on it, only works with regression functions including L, A and interactions. Manually adding predictors to bestModel via the predictors parameter is currently incompatible. In that case, and if you are primarily interested on fitting a complete data set, rather user the predict function of the stats:lm package on the ideal model solution. You than have to provide a prepared data frame with the according input variables.

Usage

```
predictRaw(
  norm,
  age,
  coefficients,
  minRaw = -Inf,
  maxRaw = Inf,
  covariate = NULL
)
```

Arguments

`norm`	The norm score, e. g. a specific T score or a vector of scores
`age`	The age value or a vector of scores
`coefficients`	The coefficients from the regression model or a cnorm model
`minRaw`	Minimum score for the results; can be used for clipping unrealistic outcomes, usually set to the lower bound of the range of values of the test (default: 0)
`maxRaw`	Maximum score for the results; can be used for clipping unrealistic outcomes usually set to the upper bound of the range of values of the test
`covariate`	In case, a covariate has been used, please specify the degree of the covariate / the specific value here.

Value

the predicted raw score or a data.frame of scores in case, lists of norm scores or age is used

See Also

Other predict: `derivationTable()`, `getNormCurve()`, `normTable()`, `predictNorm()`, `rawTable()`

Examples

```
# Prediction of single scores
normData <- prepareData(elfe)
m <- bestModel(data = normData)
predictRaw(35, 3.5, m$coefficients)

# using a cnorm object
result <- cnorm(raw = elfe$raw, group = elfe$group)
predictRaw(35, 3.5, result)

# Fitting complete data sets
fitted.values <- predict(m)

# break up contribution of each predictor variable
fitted.partial <- predict(m, type = "terms")
```

`prepareData` *Prepare data for modeling in one step (convenience method)*

Description

This is a convenience method to either load the inbuilt sample dataset, or to provide a data frame with the variables "raw" (for the raw scores) and "group" The function ranks the data within groups, computes norm values, powers of the norm scores and interactions. Afterwards, you can use these preprocessed data to determine the best fitting model.

Usage

```
prepareData(
  data = NULL,
  group = "group",
  raw = "raw",
  age = "group",
  k = 4,
  t = NULL,
  width = NA,
  weights = NULL,
  scale = "T",
  descend = FALSE,
  silent = FALSE
)
```

Arguments

`data`	data.frame with a grouping variable named 'group' and a raw score variable named 'raw'.
`group`	grouping variable in the data, e. g. age groups, grades ... Setting group = FALSE deactivates modeling in dependence of age. Use this in case you do want conventional norm tables.
`raw`	the raw scores
`age`	the continuous explanatory variable; by default set to "group"
`k`	The power parameter, default = 4
`t`	the age power parameter (default NULL). If not set, cNORM automatically uses k. The age power parameter can be used to specify the k to produce rectangular matrices and specify the course of scores per independently from k
`width`	if a width is provided, the function switches to rankBySlidingWindow to determine the observed raw scores, otherwise, ranking is done by group (default)
`weights`	Vector or variable name in the dataset with weights for each individual case. It can be used to compensate for moderate imbalances due to insufficient norm data stratification. Weights should be numerical and positive. Please use the 'computeWeights' function for this purpose.
`scale`	type of norm scale, either T (default), IQ, z or percentile (= no transformation); a double vector with the mean and standard deviation can as well, be provided f. e. c(10, 3) for Wechsler scale index point
`descend`	ranking order (default descent = FALSE): inverses the ranking order with higher raw scores getting lower norm scores; relevant for example when norming error scores, where lower scores mean higher performance
`silent`	set to TRUE to suppress messages

Value

data frame including the norm scores, powers and interactions of the norm score and grouping variable

See Also

Other prepare: `computePowers()`, `rankByGroup()`, `rankBySlidingWindow()`

Examples

```
# conducts ranking and computation of powers and interactions with the 'elfe' dataset
data.elfe <- prepareData(elfe)

# use vectors instead of data frame
data.elfe <- prepareData(raw=elfe$raw, group=elfe$group)

# variable names can be specified as well, here with the BMI data included in the package
## Not run:
data.bmi <- prepareData(CDC, group = "group", raw = "bmi", age = "age")

## End(Not run)

# modeling with only one group with the 'elfe' dataset as an example
# this results in conventional norming
data.elfe2 <- prepareData(data = elfe, group = FALSE)
m <- bestModel(data.elfe2)
```

---

prettyPrint — *Format raw and norm tables The function takes a raw or norm table, condenses intervals at the bottom and top and round the numbers to meaningful interval.*

Description

Format raw and norm tables The function takes a raw or norm table, condenses intervals at the bottom and top and round the numbers to meaningful interval.

Usage

```
prettyPrint(table)
```

Arguments

`table`	The table to format

Value

formatted table

`print.cnorm` *S3 method for printing model selection information*

Description

After conducting the model fitting procedure on the data set, the best fitting model has to be chosen. The print function shows the R2 and other information on the different best fitting models with increasing number of predictors.

Usage

```
## S3 method for class 'cnorm'
print(x, ...)
```

Arguments

`x`	The model from the 'bestModel' function or a cnorm object
`...`	additional parameters

Value

A table with information criteria

See Also

Other model: `bestModel()`, `checkConsistency()`, `cnorm.cv()`, `derive()`, `modelSummary()`, `printSubset()`, `rangeCheck()`, `regressionFunction()`, `summary.cnorm()`

`printSubset` *Print Model Selection Information*

Description

Displays R^2 and other metrics for models with varying predictors, aiding in choosing the best-fitting model after model fitting.

Usage

```
printSubset(x, ...)
```

Arguments

`x`	Model output from 'bestModel' or a cnorm object.
`...`	Additional parameters.

Value

Table with model information criteria.

See Also

Other model: bestModel(), checkConsistency(), cnorm.cv(), derive(), modelSummary(), print.cnorm(), rangeCheck(), regressionFunction(), summary.cnorm()

Examples

```
# Using cnorm object from sample data
result <- cnorm(raw = elfe$raw, group = elfe$group)
printSubset(result)
```

---

rangeCheck *Check for horizontal and vertical extrapolation*

Description

Regression model only work in a specific range and extrapolation horizontally (outside the original range) or vertically (extreme norm scores) might lead to inconsistent results. The function generates a message, indicating extrapolation and the range of the original data.

Usage

```
rangeCheck(
  object,
  minAge = NULL,
  maxAge = NULL,
  minNorm = NULL,
  maxNorm = NULL,
  digits = 3,
  ...
)
```

Arguments

object	The regression model or a cnorm object
minAge	The lower age bound
maxAge	The upper age bound
minNorm	The lower norm value bound
maxNorm	The upper norm value bound
digits	The precision for rounding the norm and age data
...	additional parameters

Value

the report

See Also

Other model: `bestModel()`, `checkConsistency()`, `cnorm.cv()`, `derive()`, `modelSummary()`, `print.cnorm()`, `printSubset()`, `regressionFunction()`, `summary.cnorm()`

Examples

```
normData <- prepareData(elfe)
m <- bestModel(normData)
rangeCheck(m)
```

---

rankByGroup — *Determine the norm scores of the participants in each subsample*

Description

This is the initial step, usually done in all kinds of test norming projects, after the scale is constructed and the norm sample is established. First, the data is grouped according to a grouping variable and afterwards, the percentile for each raw value is retrieved. The percentile can be used for the modeling procedure, but in case, the samples to not deviate too much from normality, T, IQ or z scores can be computed via a normal rank procedure based on the inverse cumulative normal distribution. In case of bindings, we use the medium rank and there are different methods for estimating the percentiles (default RankIt).

Usage

```
rankByGroup(
  data = NULL,
  group = "group",
  raw = "raw",
  weights = NULL,
  method = 4,
  scale = "T",
  descend = FALSE,
  descriptives = TRUE,
  covariate = NULL,
  na.rm = TRUE,
  silent = FALSE
)
```

Arguments

`data`	data.frame with norm sample data. If no data.frame is provided, the raw score and group vectors are directly used
`group`	name of the grouping variable (default 'group') or numeric vector, e. g. grade, setting group to FALSE cancels grouping (data is treated as one group)
`raw`	name of the raw value variable (default 'raw') or numeric vector
`weights`	Vector or variable name in the dataset with weights for each individual case. It can be used to compensate for moderate imbalances due to insufficient norm data stratification. Weights should be numerical and positive. Please use the 'computeWeights' function for this purpose.
`method`	Ranking method in case of bindings, please provide an index, choosing from the following methods: 1 = Blom (1958), 2 = Tukey (1949), 3 = Van der Warden (1952), 4 = Rankit (default), 5 = Levenbach (1953), 6 = Filliben (1975), 7 = Yu & Huang (2001)
`scale`	type of norm scale, either T (default), IQ, z or percentile (= no transformation); a double vector with the mean and standard deviation can as well, be provided f. e. c(10, 3) for Wechsler scale index points
`descend`	ranking order (default descent = FALSE): inverses the ranking order with higher raw scores getting lower norm scores; relevant for example when norming error scores, where lower scores mean higher performance
`descriptives`	If set to TRUE (default), information in n, mean, median and standard deviation per group is added to each observation
`covariate`	Include a binary covariate into the preparation and subsequently modeling, either by specifying the variable name or including the variable itself. BEWARE! Not all subsequent functions are already prepared for it. It is an experimental feature.
`na.rm`	remove values, where the percentiles could not be estimated, most likely happens in the context of weighting
`silent`	set to TRUE to suppress messages

Value

the dataset with the percentiles and norm scales per group

Remarks on using covariates

So far the inclusion of a binary covariate is experimental and far from optimized. The according variable name has to be specified in the ranking procedure and the modeling includes this in the further process. At the moment, during ranking the data are split into the according cells group x covariate, which leads to small sample sizes. Please take care to have enough cases in each combination. Additionally, covariates can lead to unstable modeling solutions. The question, if it is really reasonable to include covariates when norming a test is a decision beyond the pure data modeling. Please use with care or alternatively split the dataset into the two groups beforehand and model them separately.

See Also

rankBySlidingWindow, computePowers, computeWeights, weighted.rank

Other prepare: computePowers(), prepareData(), rankBySlidingWindow()

Examples

```
# Transformation with default parameters: RankIt and converting to T scores
data.elfe <- rankByGroup(elfe, group = "group") # using a data frame with vector names
data.elfe2 <- rankByGroup(raw=elfe$raw, group=elfe$group) # use vectors for raw score and group

# Transformation into Wechsler scores with Yu & Huang (2001) ranking procedure
data.elfe <- rankByGroup(raw = elfe$raw, group = elfe$group, method = 7, scale = c(10, 3))

# cNORM can as well be used for conventional norming, in case no group is given
d <- rankByGroup(raw = elfe$raw)
d <- computePowers(d)
m <- bestModel(d)
rawTable(0, m) # please use an arbitrary value for age when generating the tables
```

---

rankBySlidingWindow *Determine the norm scores of the participants by sliding window (experimental)*

Description

The function retrieves all individuals in the predefined age range (x +/- width/2) around each case and ranks that individual based on this individually drawn sample. This function can be directly used with a continuous age variable in order to avoid grouping. When collecting data on the basis of a continuous age variable, cases located far from the mean age of the group receive distorted percentiles when building discrete groups and generating percentiles with the traditional approach. The distortion increases with distance from the group mean and this effect can be avoided by the sliding window. Nonetheless, please ensure, that the optional grouping variable in fact represents the correct mean age of the respective age groups, as this variable is later on used for displaying the manifest data in the percentile plots.

Usage

```
rankBySlidingWindow(
  data = NULL,
  age = "age",
  raw = "raw",
  weights = NULL,
  width,
  method = 4,
  scale = "T",
  descend = FALSE,
  descriptives = TRUE,
```

```
  nGroup = 0,
  group = NA,
  covariate = NULL,
  na.rm = TRUE,
  silent = FALSE
)
```

Arguments

data	data.frame with norm sample data
age	the continuous age variable. Setting 'age' to FALSE inhibits computation of powers of age and the interactions
raw	name of the raw value variable (default 'raw')
weights	Vector or variable name in the dataset with weights for each individual case. It can be used to compensate for moderate imbalances due to insufficient norm data stratification. Weights should be numerical and positive. It can be resource intense when applied to the sliding window. Please use the 'computeWeights' function for this purpose.
width	the width of the sliding window
method	Ranking method in case of bindings, please provide an index, choosing from the following methods: 1 = Blom (1958), 2 = Tukey (1949), 3 = Van der Warden (1952), 4 = Rankit (default), 5 = Levenbach (1953), 6 = Filliben (1975), 7 = Yu & Huang (2001)
scale	type of norm scale, either T (default), IQ, z or percentile (= no transformation); a double vector with the mean and standard deviation can as well, be provided f. e. c(10, 3) for Wechsler scale index points
descend	ranking order (default descent = FALSE): inverses the ranking order with higher raw scores getting lower norm scores; relevant for example when norming error scores, where lower scores mean higher performance
descriptives	If set to TRUE (default), information in n, mean, median and standard deviation per group is added to each observation
nGroup	If set to a positive value, a grouping variable is created with the desired number of equi distant groups, named by the group mean age of each group. It creates the column 'group' in the data.frame and in case, there is already one with that name, overwrites it.
group	Optional parameter for providing the name of the grouping variable (if present; overwritten if ngroups is used)
covariate	Include a binary covariate into the preparation and subsequently modeling, either by specifying the variable name or including the variable itself. BEWARE! Not all subsequent functions are already prepared for it. It is an experimental feature.
na.rm	remove values, where the percentiles could not be estimated, most likely happens in the context of weighting
silent	set to TRUE to suppress messages

Details

In case of bindings, the function uses the medium rank and applies the algorithms already described in the rankByGroup function. At the upper and lower end of the data sample, the sliding stops and the sample is drawn from the interval min + width and max - width, respectively.

Value

the dataset with the individual percentiles and norm scores

Remarks on using covariates

So far the inclusion of a binary covariate is experimental and far from optimized. The according variable name has to be specified in the ranking procedure and the modeling includes this in the further process. At the moment, during ranking the data are split into the according degrees of the covariate and the ranking is done separately. This may lead to small sample sizes. Please take care to have enough cases in each combination. Additionally, covariates can lead to unstable modeling solutions. The question, if it is really reasonable to include covariates when norming a test is a decision beyond the pure data modeling. Please use with care or alternatively split the dataset into the two groups beforehand and model them separately.

See Also

rankByGroup, computePowers, computeWeights, weighted.rank, weighted.quantile

Other prepare: computePowers(), prepareData(), rankByGroup()

Examples

```
## Not run:
# Transformation using a sliding window
data.elfe2 <- rankBySlidingWindow(relfe, raw = "raw", age = "group", width = 0.5)

# Comparing this to the traditional approach should give us exactly the same
# values, since the sample dataset only has a grouping variable for age
data.elfe <- rankByGroup(elfe, group = "group")
mean(data.elfe$normValue - data.elfe2$normValue)

## End(Not run)
```

---

rawTable — *Create a table with norm scores assigned to raw scores for a specific age based on the regression model*

Description

This function is comparable to 'normTable', despite it reverses the assignment: A table with raw scores and the according norm scores for a specific age based on the regression model is generated. This way, the inverse function of the regression model is solved numerically with brute force. Please specify the range of raw values, you want to cover. With higher precision and smaller stepping, this function becomes computational intensive. In case a confidence coefficient (CI, default .9) and the reliability is specified, confidence intervals are computed for the true score estimates, including a correction for regression to the mean (Eid & Schmidt, 2012, p. 272).

Usage

```
rawTable(
  A,
  model,
  minRaw = NULL,
  maxRaw = NULL,
  minNorm = NULL,
  maxNorm = NULL,
  step = 1,
  covariate = NULL,
  monotonuous = TRUE,
  CI = 0.9,
  reliability = NULL,
  pretty = TRUE
)
```

Arguments

`A`	the age, either single value or vector with age values
`model`	The regression model or a cnorm object
`minRaw`	The lower bound of the raw score range
`maxRaw`	The upper bound of the raw score range
`minNorm`	Clipping parameter for the lower bound of norm scores (default 25)
`maxNorm`	Clipping parameter for the upper bound of norm scores (default 25)
`step`	Stepping parameter for the raw scores (default 1)
`covariate`	In case, a covariate has been used, please specify the degree of the covariate / the specific value here.
`monotonuous`	corrects for decreasing norm scores in case of model inconsistencies (default)
`CI`	confidence coefficient, ranging from 0 to 1, default .9
`reliability`	coefficient, ranging between 0 to 1
`pretty`	Format table by collapsing intervals and rounding to meaningful precision

Value

either data.frame with raw scores and the predicted norm scores in case of simple A value or a list of norm tables if vector of A values was provided

References

Eid, M. & Schmidt, K. (2012). Testtheorie und Testkonstruktion. Hogrefe.

See Also

normTable

Other predict: derivationTable(), getNormCurve(), normTable(), predictNorm(), predictRaw()

Examples

```
# Generate cnorm object from example data
cnorm.elfe <- cnorm(raw = elfe$raw, group = elfe$group)
# generate a norm table for the raw value range from 0 to 28 for the time point month 7 of grade 3
table <- rawTable(3 + 7 / 12, cnorm.elfe, minRaw = 0, maxRaw = 28)

# generate several raw tables
table <- rawTable(c(2.5, 3.5, 4.5), cnorm.elfe, minRaw = 0, maxRaw = 28)

# additionally compute confidence intervals
table <- rawTable(c(2.5, 3.5, 4.5), cnorm.elfe, minRaw = 0, maxRaw = 28, CI = .9, reliability = .94)
```

---

regressionFunction *Regression function*

Description

The method builds the regression function for the regression model, including the beta weights. It can be used to predict the raw scores based on age and location.

Usage

```
regressionFunction(model, raw = NULL, digits = NULL)
```

Arguments

model	The regression model from the bestModel function or a cnorm object
raw	The name of the raw value variable (default 'raw')
digits	Number of digits for formatting the coefficients

Value

The regression formula as a string

See Also

Other model: bestModel(), checkConsistency(), cnorm.cv(), derive(), modelSummary(), print.cnorm(), printSubset(), rangeCheck(), summary.cnorm()

Examples

```
result <- cnorm(raw = elfe$raw, group = elfe$group)
regressionFunction(result)
```

---

simMean *Simulate mean per age*

Description

Simulate mean per age

Usage

```
simMean(age)
```

Arguments

age	the age variable

Value

return predicted means

Examples

```
## Not run:
x <- simMean(a)

## End(Not run)
```

---

simSD *Simulate sd per age*

Description

Simulate sd per age

Usage

```
simSD(age)
```

Arguments

age	the age variable

Value

return predicted sd

Examples

```
## Not run:
x <- simSD(a)

## End(Not run)
```

`simulateRasch` *Simulate raw test scores based on Rasch model*

Description

For testing purposes only: The function simulates raw test scores based on a virtual Rasch based test with n results per age group, an evenly distributed age variable, items.n test items with a simulated difficulty and standard deviation. The development trajectories over age group are modeled by a curve linear function of age, with at first fast progression, which slows down over age, and a slightly increasing standard deviation in order to model a scissor effects. The item difficulties can be accessed via $theta and the raw data via $data of the returned object.

Usage

```
simulateRasch(
  data = NULL,
  n = 100,
  minAge = 1,
  maxAge = 7,
  items.n = 21,
  items.m = 0,
  items.sd = 1,
  Theta = "random",
  width = 1
)
```

Arguments

`data`	data.frame from previous simulations for recomputation (overrides n, minAge, maxAge)
`n`	The sample size per age group
`minAge`	The minimum age (default 1)
`maxAge`	The maximum age (default 7)
`items.n`	The number of items of the test
`items.m`	The mean difficulty of the items

items.sd	The standard deviation of the item difficulty
Theta	irt scales difficulty parameters, either "random" for drawing a random sample, "even" for evenly distributed or a set of predefined values, which then overrides the item.n parameters
width	The width of the window size for the continuous age per group; +- 1/2 width around group center on items.m and item.sd; if set to FALSE, the distribution is not drawn randomly but normally nonetheless

Value

a list containing the simulated data and thetas

data the data.frame with only age, group and raw

sim the complete simulated data with item level results

theta the difficulty of the items

Examples

```
# simulate data for a rather easy test (m = -1.0)
sim <- simulateRasch(n=150, minAge=1,
                     maxAge=7, items.n = 30, items.m = -1.0,
                     items.sd = 1, Theta = "random", width = 1.0)

# Show item difficulties
mean(sim$theta)
sd(sim$theta)
hist(sim$theta)

# Plot raw scores
boxplot(raw~group, data=sim$data)

# Model data
data <- prepareData(sim$data, age="age")
model <- bestModel(data, k = 4)
printSubset(model)
plotSubset(model, type=0)
```

---

standardizeRakingWeights

Function for standardizing raking weights Raking weights get divided by the smallest weight. Thereby, all weights become larger or equal to 1 without changing the ratio of the weights to each other.

Description

Function for standardizing raking weights Raking weights get divided by the smallest weight. Thereby, all weights become larger or equal to 1 without changing the ratio of the weights to each other.

Usage

```
standardizeRakingWeights(weights)
```

Arguments

`weights`	Raking weights computed by computeWeights()

Value

the standardized weights

`summary.cnorm` *S3 method for printing the results and regression function of a cnorm model*

Description

S3 method for printing the results and regression function of a cnorm model

Usage

```
## S3 method for class 'cnorm'
summary(object, ...)
```

Arguments

`object`	A regression model or cnorm object
`...`	additional parameters

Value

A report on the regression function, weights, R2 and RMSE

See Also

Other model: `bestModel()`, `checkConsistency()`, `cnorm.cv()`, `derive()`, `modelSummary()`, `print.cnorm()`, `printSubset()`, `rangeCheck()`, `regressionFunction()`

`weighted.quantile` *Weighted quantile estimator*

Description

Computes weighted quantiles (code from Andrey Akinshin (2023) "Weighted quantile estimators" arXiv:2304.07265 [stat.ME] Code made available via the CC BY-NC-SA 4.0 license) on the basis of either the weighted Harrell-Davis quantile estimator or an adaption of the type 7 quantile estimator of the generic quantile function in the base package. Please provide a vector with raw values, the pobabilities for the quantiles and an additional vector with the weight of each observation. In case the weight vector is NULL, a normal quantile estimation is done. The vectors may not include NAs and the weights should be positive non-zero values. Please draw on the computeWeights() function for retrieving weights in post stratification.

Usage

```
weighted.quantile(x, probs, weights = NULL, type = "Harrell-Davis")
```

Arguments

x	A numerical vector
probs	Numerical vector of quantiles
weights	A numerical vector with weights; should have the same length as x
type	Type of estimator, can either be "inflation", "Harrell-Davis" using a beta function to approximate the weighted percentiles (Harrell & Davis, 1982) or "Type7" (default; Hyndman & Fan, 1996), an adaption of the generic quantile function in R, including weighting. The inflation procedure is essentially a numerical, non-parametric solution that gives the same results as Harrel-Davis. It requires less ressources with small datasets and always finds a solution (e. g. 1000 cases with weights between 1 and 10). If it becomes too resource intense, it switches to Harrell-Davis automatically. Harrel-Davis and Type7 code is based on the work of Akinshin (2023).

Value

the weighted quantiles

References

1. Harrell, F.E. & Davis, C.E. (1982). A new distribution-free quantile estimator. Biometrika, 69(3), 635-640.
2. Hyndman, R. J. & Fan, Y. (1996). Sample quantiles in statistical packages, American Statistician 50, 361–365.
3. Akinshin, A. (2023). Weighted quantile estimators arXiv:2304.07265 [stat.ME]

See Also

weighted.quantile.inflation, weighted.quantile.harrell.davis, weighted.quantile.type7

`weighted.quantile.harrell.davis`
Weighted Harrell-Davis quantile estimator

Description

Computes weighted quantiles; code from Andrey Akinshin (2023) "Weighted quantile estimators" arXiv:2304.07265 [stat.ME] Code made available via the CC BY-NC-SA 4.0 license

Usage

```
weighted.quantile.harrell.davis(x, probs, weights = NULL)
```

Arguments

`x`	A numerical vector
`probs`	Numerical vector of quantiles
`weights`	A numerical vector with weights; should have the same length as x. If no weights are provided (NULL), it falls back to the base quantile function, type 7

Value

the quantiles

`weighted.quantile.inflation`
Weighted quantile estimator through case inflation

Description

Applies weighted ranking numerically by inflating cases according to weight. This function will be resource intensive, if inflated cases get too high and in this cases, it switches to the parametric Harrell-Davis estimator.

Usage

```
weighted.quantile.inflation(
  x,
  probs,
  weights = NULL,
  degree = 3,
  cutoff = 1e+07
)
```

Arguments

`x`	A numerical vector
`probs`	Numerical vector of quantiles
`weights`	A numerical vector with weights; should have the same length as x.
`degree`	power parameter for case inflation (default = 3, equaling factor 1000) If no weights are provided (NULL), it falls back to the base quantile function, type 7
`cutoff`	stop criterion for the sum of standardized weights to switch to Harrell-Davis, default = 1000000

Value

the quantiles

`weighted.quantile.type7` *Weighted type7 quantile estimator*

Description

Computes weighted quantiles; code from Andrey Akinshin (2023) "Weighted quantile estimators" arXiv:2304.07265 [stat.ME] Code made available via the CC BY-NC-SA 4.0 license

Usage

```
weighted.quantile.type7(x, probs, weights = NULL)
```

Arguments

`x`	A numerical vector
`probs`	Numerical vector of quantiles
`weights`	A numerical vector with weights; should have the same length as x. If no weights are provided (NULL), it falls back to the base quantile function, type 7

Value

the quantiles

`weighted.rank` *Weighted rank estimation*

Description

Conducts weighted ranking on the basis of sums of weights per unique raw score. Please provide a vector with raw values and an additional vector with the weight of each observation. In case the weight vector is NULL, a normal ranking is done. The vectors may not include NAs and the weights should be positive non-zero values.

Usage

```
weighted.rank(x, weights = NULL)
```

Arguments

`x`	A numerical vector
`weights`	A numerical vector with weights; should have the same length as x

Value

the weighted absolute ranks

Index

∗ **Body Mass Index growth curves weight height**
CDC, 12
∗ **datasets**
CDC, 12
elfe, 25
epm, 26
life, 30
mortality, 31
ppvt, 44
∗ **life expectancy**
life, 30
∗ **model**
bestModel, 3
checkConsistency, 13
cnorm.cv, 17
derive, 24
modelSummary, 31
print.cnorm, 51
printSubset, 51
rangeCheck, 52
regressionFunction, 59
summary.cnorm, 63
∗ **mortality at birth**
mortality, 31
∗ **plot**
plot.cnorm, 34
plotDensity, 35
plotDerivative, 36
plotNorm, 37
plotNormCurves, 38
plotPercentiles, 40
plotPercentileSeries, 41
plotRaw, 42
plotSubset, 43
∗ **predict**
derivationTable, 23
getNormCurve, 28
normTable, 32
predictNorm, 46
predictRaw, 47
rawTable, 57
∗ **prepare**
computePowers, 20
prepareData, 48
rankByGroup, 53
rankBySlidingWindow, 55
∗ **reading comprehension**
elfe, 25
∗ **simulated data 1PL IRT**
epm, 26
∗ **vocabulary acquisition development receptive**
ppvt, 44

bestModel, 3, *14*, *19, 20*, *25*, *31*, *51–53*, *59*, *63*
betaByGroup, 5
betaCoefficients, 6
betaContinuous, 7
betaNormTable, 8
betaTable, 9
buildFunction, 10

calcPolyInL, 10
calcPolyInLBase, 11
calcPolyInLBase2, 11
CDC, 12
checkConsistency, *4*, 13, *19*, *25*, *31*, *51–53*, *59*, *63*
checkWeights, 14
cnorm, 15
cnorm.cv, *4*, *14*, 17, *25*, *31*, *51–53*, *59*, *63*
cNORM.GUI, 20
computePowers, 20, *50*, *55*, *57*
computeWeights, 22

derivationTable, 23, *29*, *33*, *47, 48*, *59*
derive, *4*, *14*, *19*, 24, *31*, *51–53*, *59*, *63*

elfe, 25

epm, 26

getGroups, 27
getNormCurve, *24*, 28, *33*, *47, 48*, *59*
getNormScoreSE, 29

life, 30

modelSummary, *4*, *14*, *19*, *25*, 31, *51–53*, *59*, *63*
mortality, 31

normTable, *24*, *29*, 32, *47, 48*, *59*

plot.cnorm, 34, *36–39*, *41–44*
plotCnorm, 35
plotDensity, *34*, 35, *37–39*, *41–44*
plotDerivative, *34*, *36*, 36, *38, 39*, *41–44*
plotNorm, *34*, *36, 37*, 37, *39*, *41–44*
plotNormCurves, *34*, *36–38*, 38, *41–44*
plotPercentiles, *34*, *36–39*, 40, *42–44*
plotPercentileSeries, *34*, *36–39*, *41*, 41, *43, 44*
plotRaw, *34*, *36–39*, *41, 42*, 42, *44*
plotSubset, *34*, *36–39*, *41–43*, 43
ppvt, 44
predictBeta, 45
predictNorm, *24*, *29*, *33*, 46, *48*, *59*
predictRaw, *24*, *29*, *33*, *47*, 47, *59*
prepareData, *21*, 48, *55*, *57*
prettyPrint, 50
print.cnorm, *4*, *14*, *19*, *25*, *31*, 51, *52, 53*, *59*, *63*
printSubset, *4*, *14*, *19*, *25*, *31*, *51*, 51, *53*, *59*, *63*

rangeCheck, *4*, *14*, *19*, *25*, *31*, *51, 52*, 52, *59*, *63*
rankByGroup, *21*, *50*, 53, *57*
rankBySlidingWindow, *21*, *50*, *55*, 55
rawTable, *24*, *29*, *33*, *47, 48*, 57
regressionFunction, *4*, *14*, *19*, *25*, *31*, *51–53*, 59, *63*

simMean, 60
simSD, 60
simulateRasch, 61
standardizeRakingWeights, 62
summary.cnorm, *4*, *14*, *19*, *25*, *31*, *51–53*, *59*, 63

weighted.quantile, 64
weighted.quantile.harrell.davis, 65
weighted.quantile.inflation, 65
weighted.quantile.type7, 66
weighted.rank, 67